

FastMask: Segment Multi-scale Object Candidates in One Shot

Hexiang Hu^{*†}
University of California, Los Angeles
Los Angeles, CA
hexiang.frank.hu@gmail.com

Shiyi Lan^{*†}
Fudan University
Shanghai, China
sylan14@fudan.edu.cn

Yuning Jiang
Megvii Inc.
Beijing, China
jyn@megvii.com

Zhimin Cao
Megvii Inc.
Beijing, China
czm@megvii.com

Fei Sha
University of California, Los Angeles
Los Angeles, CA
feisha@cs.ucla.edu

Abstract

Objects appear to scale differently in natural images. This fact requires methods dealing with object-centric tasks (e.g. object proposal) to have robust performance over scale variances of objects. In the paper we present a novel segment proposal framework, namely FastMask, which takes advantage of the hierarchical structure in deep convolutional neural network to segment multi-scale objects in one shot. Innovatively, we generalize segment proposal network into three different functional components (body, neck and head). We further propose a weight-shared residual neck module as well as a scale-tolerant attentional head module for multi-scale training and efficient one-shot inference. On MS COCO benchmark, the proposed FastMask outperforms all state-of-the-art segment proposal methods in average recall while keeping 2~5 times faster. More impressively, with a slight tradeoff in accuracy, FastMask can segment objects in near real time (~13 fps) at 800×600 resolution images, highlighting its potential in practical applications. Our implementation is available on <https://github.com/voidrank/FastMask>.

1. Introduction

With the goal to propose object candidates from images, object proposal is considered as the first and fundamental step in object detection task [8, 22, 1, 14, 9, 24]. As the domain rapidly progressed, a renewed interest in object segment proposal has received intensive attentions [6, 17, 18, 5, 2]. Compared to traditional object proposal methods, segment proposal algorithms are expected to output a pixel-

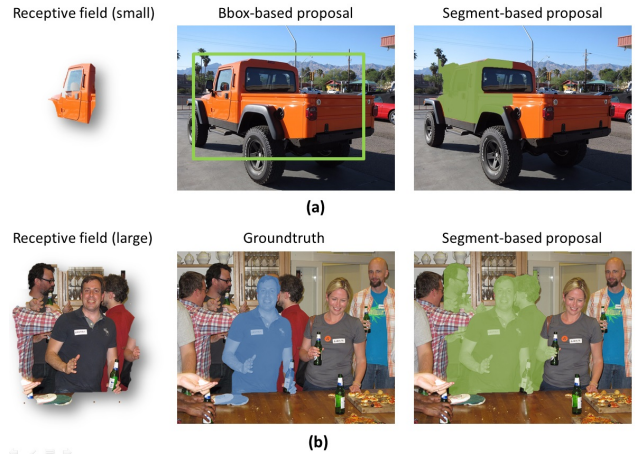


Figure 1. (a) With a very small receptive field, such as perceiving only a window of a car, bbox-based proposer could still make an estimation while it becomes impossible for the segment-based proposer to estimate the contour line of car; (b) With a very large receptive field, a segment-based proposer could be distracted by the noises in backgrounds, e.g. other people surrounding the target person.

wise segment instead of a bounding box for each individual object candidate. From this perspective, segment proposal inherits from both object proposal and image segmentation and takes a step further towards simultaneous detection and segmentation, which brings more difficult problems to overcome.

Among all the problems, how to handle the objects appearing in large scale variances is the most challenging one. Though scale variance also exists in bounding-box-based (bbox-based in short) object proposal, it becomes more serious for segment proposal task. It is due to the fact that segment proposal raises the need for a highly matched receptive field with the varying scales of objects. On one hand, when the receptive field of object proposer is smaller than

^{*}Equal contribution.

[†]Work was done during their internships at Megvii Inc.

the object, such as perceiving only a window of a car, the bbox-based proposer could still make an estimation on the entire bounding box of the car with prior knowledge. However, it is much more difficult for the segment-based proposer to estimate the contour line of a car given only a view of its window. On the other hand, very large receptive field will probably introduce noises from backgrounds and hence result in the incorrect boundaries of pixel-level segments. For example, a segment-based proposer will probably be distracted by other people standing nearby the target person, and thus it tends to generate a coarse mask covering all people in its receptive field as a whole while a fine mask for the target individual is demanded in deed (see Figure 1). As a consequence, the scale variance problem badly affects the segment proposal performance in both segmentation fineness and proposal recall. In other words, how to segment the object candidates in varying scales plays a critical role in segment proposal methods.

In general, the existing methods [6, 17, 18, 5, 2] could be mainly divided into two categories by their ways to handle scale variance. The first category of methods [6, 2] uses extra bbox-based object proposal or object detection results as initial inputs. However, its effectiveness and efficiency are highly dependent on the accuracy and speed of the pre-processing proposal methods, respectively. The second category of methods [17, 18, 5] adopts the image pyramid strategy, in which the original image is zoomed out and fed into a fixed-scale object proposer repeatedly to segment the objects in different scales (see Figure 1(a)). However, this category of methods has to face a common dilemma: a densely sampled image pyramid will become the computational bottleneck of the whole framework; nevertheless, reducing the number of the scales in image pyramid will lead to performance degradation in segmentation accuracy and proposal recall. That means, these methods could not provide satisfactory accuracy and speed at the same time. In addition, with the observation that the largest image has already contained all information of an image pyramid, we argue that taking one single image should be enough to capture all multi-scale objects in it.

Therefore, in this paper we aim to address the scale variance problem in segment proposal by leveraging hierarchical structure of convolutional neural networks (CNN). Innovatively, we generalize segment proposal network into three different functional components, namely *body*, *neck* and *head*. Similar to [17, 18], the body and head module are responsible for extracting semantic feature maps from original images and decoding segmentation masks from feature maps, respectively; the difference is that we introduce the concept of neck module, whose job is to recurrently zoom out the feature maps extracted by the body module into feature pyramids, and then feed the feature pyramids into the head module for multi-scale inference. Our main contribu-

tions are summarized as:

- First, we learn a novel weight-shared residual neck module to zoom out feature maps of CNN while preserving calibrated feature semantics, which enables efficient multi-scale training and inference.
- Next, we propose a novel scale-tolerant head module which takes advantage of attention model and significantly reduces the impact of background noises caused by unmatched receptive fields.
- Finally, together with all those modules, we make up a framework capable for one-shot segment proposal, namely FastMask. We evaluate the proposed framework on MS COCO benchmark [16] and it achieves the state-of-the-art results in accuracy while running in near real time.

2. Related Work

Bbox-based object proposal. Most of the bbox-based object proposal methods rely on the dense sliding window on image pyramid. In EdgeBox [25] and Bing [4], the edge feature is used to make the prediction for each sliding window while the gradient feature is used in [24]. More recently, with the effects of CNNs, DeepBox [15] trains a network to re-rank the proposals generated by EdgeBox, while MultiBox [7] generates the proposals from convolutional feature maps directly. In Faster RCNN [19], a region proposal network (RPN) is proposed, in which anchor boxes are adopted to handle object candidates in varying scales.

Segment-based object proposal. Segments proposal algorithms aim to find diverse regions in an image which are likely to contain objects. Traditional segment proposal methods such as SelectiveSearch [22], MCG [1] and Geodesic [14] first over-segment image into super pixels and then merge the super pixels in a bottom-up fashion. Inspired by the success of CNNs in image segmentation [20, 3, 23], previous works [6, 2] perform segmentation on the bbox-based object proposal results to obtain object segments. As the state-of-the-arts, DeepMask [17] proposes a body-head structure to decode object masks from CNN feature maps, and SharpMask [18] further adds a backward branch to refine the masks. In InstanceFCN [5], a translation-aware head module is employed to improve the segment accuracy. However, all these methods rely on an image pyramid during inference, which limits their application in practice.

3. Review of DeepMask

DeepMask [17] is considered as the representative of the CNN-based segment proposal methods, in which a body-head structure is proposed. In this section, to have a better understanding on the paradigm of previous methods, we

first give a review on both training and inference phases of DeepMask in details.

Training phase. DeepMask is trained to predict a segmentation mask as well as a confidence score given a fixed-scale image patch. For training, an image patch is assigned to be a positive sample if it satisfies the *object-centric constrain* [17]; otherwise the patch is assigned to be negative. Then all the patches are cropped and resized into a fixed scale (e.g., 224×224). The fixed-scale patches are fed into the body module of DeepMask, which is made up by a fully convolutional network (FCN) to extract semantic feature map from each patch. Finally, the head module (e.g., two branches of fully connected layers) decodes the confidence score as well as the segmentation mask from the feature map.

Inference phase. During full image inference, DeepMask applies the trained model densely at each location and scale. As shown in Figure 2(a), at first the input image is resized repeatedly into an image pyramid. Next, for efficiency the body module of DeepMask extracts a full feature map from each resized image since the convolutional computation should be shared. Finally the fixed-size sliding window (e.g., 14×14 if the downscale factor is 16) is performed on multi-scale feature maps, and the head module decodes the confidence score and mask for each sliding window.

From the review above we can see that a densely sampled image pyramid is needed during inference since DeepMask is trained on the fixed-scale cropped patches. However, as the convolutional computation over images in different scales is not shared, the image pyramid has already become the computational bottleneck in the paradigm of DeepMask-like segment proposal methods.

To overcome the inefficiency brought by image pyramid, we propose a new paradigm that enables efficient multi-scale training and inference. As shown in Figure 2(b), in this new paradigm we inherits the body-head structure and introduce a new component called *neck*. This neck component could be used on the feature map extracted from the body module and zoom it out into feature pyramid while preserving feature semantics. Then a head module is applied on the pyramid of feature maps to decode object segments in different scales. With the proposed body-neck-head structure, we could save the redundant convolutional computation and make efficient use of information to perform segment proposal in one shot. We refer this as **one-shot segment proposal paradigm** and derive our proposed segment proposal framework in section 4.

4. Our Approach

In this section, we introduce our approach step-by-step. First we overview the proposed architecture, namely FastMask, to give a concrete idea about our body-neck-head structure. We explain our entire network by illustrating the

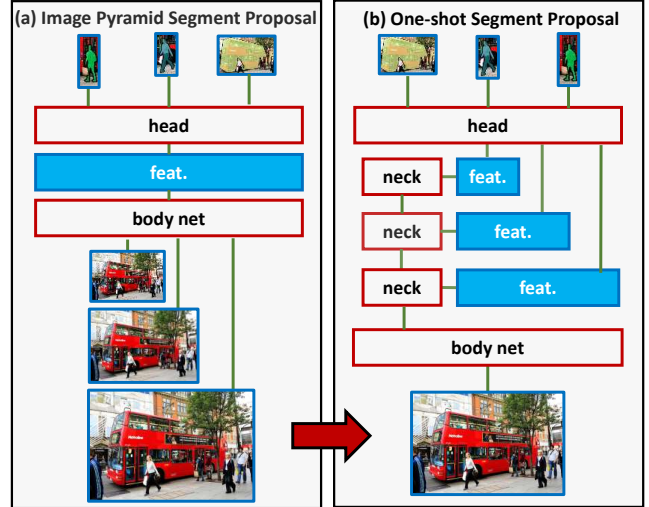


Figure 2. Comparison between image pyramid based proposal and our one-shot proposal.

data flow from input image to output object segments. Next we study the different designs of the neck module, including both the non-parametric and parametric necks. Furthermore, we present a novel head module that enables scale-tolerant segmentation mask decoding by taking advantage of the attention model.

4.1. Network Architecture

We present our network architecture which follows the one-shot paradigm developed in previous discussion (see Figure 3). In detail, the body network extracts semantic feature from input image. With this base feature map, a shared neck module is applied recursively at it to build feature maps with different scales. These feature maps are then input to a one-by-one convolution to reduce their feature dimensionality. Then we extract dense sliding windows from those feature maps and do a batch normalization across all windows to calibrate and redistribute window feature maps. Note that with a feature map downsampled by factor m , a sliding window of size (k, k) corresponds to a patch of $(m \times k, m \times k)$ at original image. Finally, a unified head module is used to decode these window features and produce the output confidence score as well as object mask.

Our approach is general and can be easily adopted to any existing CNN architecture (e.g. VGGNet [21], ResNet [10]), by replacing its fully connected layers as well as some convolutional and pooling layers on the top by the neck and head module. The reason for removing those top convolutional and pooling layers is to keep feature map in a feasible size, so that small objects can still corresponds to a notable region in feature map.

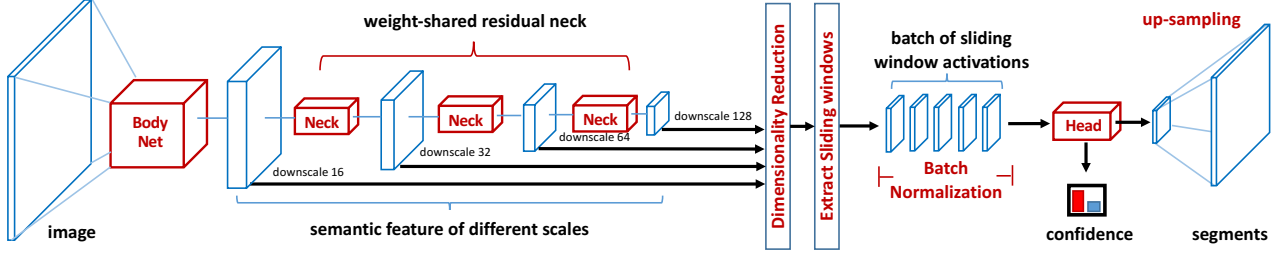


Figure 3. An overview of the proposed FastMask architecture.

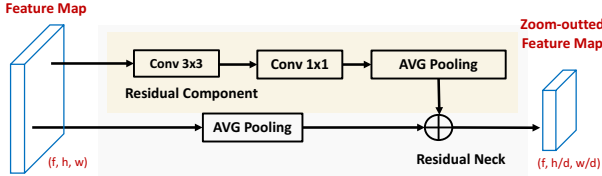


Figure 4. Illustration of the residual neck. In the residual neck, we augment the average pooling neck with a learnable residual component.

4.2. Residual Neck

There are both non-parametric and parametric methods for encoding feature pyramid. To zoom out feature map, a natural and simple choice for neck is non-parametric pooling. Both max pooling and average pooling are widely used components in the modern CNN architecture on recognition and detection. However, in our scenario that we want calibrated semantic feature for unified decoding, these pooling necks fail by their natural. In this section, we discuss about the several choices of the neck module and compare them empirically.

Max pooling neck. Max pooling produces uncalibrated feature in encoding. Suppose we have a spatial grid of feature, max pooling takes the max response in a local region and makes it the representative on downsampled feature. This process pushes the mean of downsampled feature higher than original. As we progressively apply max pooling to encode feature pyramid, the top feature maps would have significantly larger mean than the bottom ones. This would bring difficulty for the head module to decode.

Average pooling neck. On the contrary, average pooling smoothes out discriminative feature during encoding. Compared to max pooling, average pooling does not push up the mean of feature because it computes the mean of a local region as the representative for downsampled feature. Although this helps to keep features of different scales calibrated, it would smooth out discriminative feature and make the top feature maps appear to be blurry. The lost of discriminative feature makes the head module suffer from distinguishing the object to its background.

Feed-forward neck. To reduce the side-effects brought in

Method	AR@100	AR ^S @100	AR ^M @100	AR ^L @100
Avg-Pooling	27.9	11.5	36.9	43.9
Max-Pooling	27.8	11.1	36.8	44.2
Feed-Forward	27.1	10.8	35.8	43.4
Residual	29.3	11.7	38.3	47.2

Table 1. Comparison on different designs of the neck modules (on COCO benchmark). VGGNet [21] is used as body network for all the necks.

by non-parametric necks, we propose to learn parametric necks that preserve feature semantics. One naive parametric choice is to learn a feed-forward neck which uses convolutional and average pooling layers to zoom out feature maps. However, the feed-forward neck faces the gradient vanishing effect [11] as the number of scales increases. In addition, the feature semantic is probably to be changed since the feature maps on the top go through more convolutional operations than the bottom ones.

Residual neck. Inspired by bottle-neck connection in [10], we design to learn a residual neck as in figure 4. We augment the non-parametric average pooling with a parametric residual component (using the same structure as in the feed-forward neck, a 3×3 convolutional layer followed by a 1×1 one) to zoom out feature maps, in order to reduce the smooth effect of average pooling as well as preserve feature semantics.

Comparison. To verify the effectiveness of the mentioned necks, we did empirical evaluations and report their performance in table 1. Here we report overall AR@100 and AR@100 for objects in different sizes (details given in section 6). The results show that the residual neck component beats all other necks in terms of average recall. Note that we obtain a large margin in average recall for the objects in large scale, which are decoded from the top feature maps. This verifies the effectiveness of the residual neck in encoding feature pyramid.

4.3. Attentional Head

Following [17, 18], we use a simple combination of convolutional layer and fully connected layer to assemble a head module for decoding mask and confidence. How-

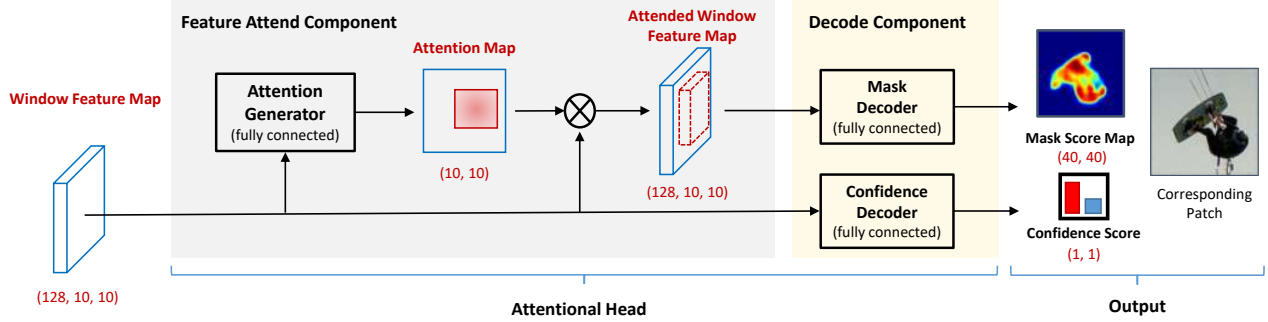


Figure 5. Details of the attentional head. It presents the data flow starting from one feature map to the decoded mask inside sliding windows. The notations in round brackets indicate the dimensionality of the data.

ever, in the context of feature pyramid decoding, we found that simply applying this head would result in a suboptimal performance. A reason for the effect is that, comparing to original DeepMask [17] framework, our feature pyramid is sparser in scales. For example, after we apply our neck module for one time, the feature map is downscaled by a factor of two, which means that the scale gap between two adjacent feature maps is two (while the scale gap in DeepMask is $2^{0.5}$). The sparse feature pyramid raises the probability that there exists no suitable feature maps for an object to decode, and also raises the risk of introducing background noises when the object is decoded from an unsuitable feature map with too larger receptive field.

Such observations motivate us to propose a novel head module that learns to attend salient region during decoding. With the capability of paying attention to the salient region, a decoding head could reduce the noises from the backgrounds of a sliding window and thus produce high quality segmentation results when the receptive field is unmatched with the scale of object. Note that the salient region attention also has the tolerance to shift disturbance (*i.e.* when the object is not centered in the sliding window), which further improves the robustness of a head module.

Figure 5 gives the detailed implementation of our attentional head. Given the feature map of a sliding window as the input, we first generate a spatial attention through a fully connected layer, which takes the entire window feature to generate the attention score for each spatial location on the feature map. The spatial attention is then applied to window feature map via the element-wise multiplication across channels. Such operation enables the head module to enhance features on the salient region, where is supposed to be the rough location of the target object. Finally, the enhanced feature map will be fed into a fully connected layer

Method	AR@10	AR@100	AR@1k
Standard Head	12.7	24.8	33.2
Attentional Head	15.2	29.3	38.6

Table 2. Comparison of different head modules on the COCO benchmark. VGGNet [21] is used as the body network.

to decode the segmentation mask of the object.

Comparison. To verify the effectiveness of the proposed attentional head, we did an experimental comparison between FastMask with a standard head and FastMask with an attentional head, as reported in table 2. From the table we can see that with the tolerance to scale and shift disturbance, the attentional head significantly improves the segment proposal accuracy.

5. Implementation Details

5.1. Training

The key difference between training FastMask and standard DeepMask [17], is that we train our network fully convolutionally on single scale image instead of multi-scale patches. To enable this training scheme, we need to assign ground truth objects to sliding-windows at different feature maps, so our head can learn to generate attention and decode mask as well as score confidence. Once we have all the ground truth window-segments pair ready, we can train our network through optimizing learning objective and back-propagate gradients. We also crafted body network to produce two stream of features and enrich the scale density of feature pyramid.

Segment matching. During training, we need to determine which sliding window a ground truth segment belongs to and train the network accordingly. For each ground truth segment, we assign them to a sliding window if (i) it fully

contains this object (ii) the object fits into the scale range of [0.4, 0.8] with regard to the window and (iii) the object is roughly centered in the window (object center in central 10×10 rectangle region of window). Once an object is assigned to a window, we crop the segmentation mask as segmentation ground truth and extract the ground truth bounding box inside window to form attention ground truth. Note that we can always derive window attention from ground-truth bounding box or mask, and do not require any extra annotation.

Learning objective. The overall objective function of FastMask is a weighted sum of the confidence loss(\mathcal{L}_{conf}), segmentation loss(\mathcal{L}_{seg}) and region attention loss(\mathcal{L}_{att}). Note that c, a, s stands for ground truth label for confidence, region attention and segmentation mask, while $\hat{c}, \hat{a}, \hat{s}$ stands for corresponding prediction.

$$\mathcal{L}(c, a, s) = \frac{1}{N} \sum_k^N [\mathcal{L}_{conf}(c_k, \hat{c}_k) + \mathbb{1}(c_k) \cdot (\mathcal{L}_{seg}(s_k, \hat{s}_k) + \mathcal{L}_{att}(a_k, \hat{a}_k))] \quad (1)$$

Here $\mathbb{1}(c_k)$ is an indicator function which returns 1 if c_k is true and 0 otherwise. Equation 1 indicates that we only back-propagate gradients when $c_k = 1$. It is critical to get good performance by computing \mathcal{L}_{seg} and \mathcal{L}_{att} only with positive object samples. We normalize this weighted sum with the total number of sliding windows across mini-batches. For each loss components, we compute the cross entropy function between the prediction and ground truth as following:

$$\mathcal{L}_{conf}(c, \hat{c}) = -\mathcal{E}(s_{i,j}, \hat{s}_{i,j}) \quad (2)$$

$$\mathcal{L}_{seg}(s, \hat{s}) = -\left[\frac{1}{w \cdot h} \sum_{i,j}^{h,w} (\mathcal{E}(s_{i,j}, \hat{s}_{i,j}))\right] \quad (3)$$

$$\mathcal{L}_{att}(a, \hat{a}) = -\left[\frac{1}{w \cdot h} \sum_{i,j}^{h,w} (\mathcal{E}(a_{i,j}, \hat{a}_{i,j}))\right] \quad (4)$$

For \mathcal{L}_{seg} and \mathcal{L}_{att} , we normalize spatially across the window to balance the gradients between three loss components. $\mathcal{E}(y, \hat{y})$ is a standard binary cross entropy function with sigmoid activation function (denoted by $\sigma(y)$), in the following form.

$$\mathcal{E}(y, \hat{y}) = y \cdot \log(\sigma(\hat{y})) + (1 - y) \cdot \log(1 - \sigma(\hat{y})) \quad (5)$$

Two-stream network. Although training fully convolutionally could benefit our inference, it has some short-come as feature scales are sparse (e.g. always multiple of 2) in the single stream fashion. To make our method with big scale density, we craft the body network to branch in the middle through applying a middle pooling layer with different

strides (e.g. 2 and 3 in our implementation) and feed these differently scaled feature to top set of convolutions. This produce a body network with two streams on the top, which generates feature maps of different sizes. In our practice, we branch a 2×2 pooling on the feature downsampled by 8 to generate feature downsampled by factors of 16 and 24, and input these feature to the shared top convolutions. Then we apply our neck and head modules on these two streams to produce object segments in different scales. This technique adds more scales of feature, helps FastMask to be more robust to scale difference, but introduce limited extra computation. Note that we do not add any new parameters for learning through this branching technique.

Optimization. We optimize the objective by standard stochastic gradient descent (SGD) with batch size equals 1, momentum equals 0.9 and weight decay equals 0.00005. We train our network for approximately 15 epochs and choose the best models through a different subset of COCO validation set. Following the practice of [19, 8], we balance positive and negative samples by a certain ratio (e.g. roughly 1:1 in our case) after collecting all sliding-windows in training. In our practice, due to the limitation of GPU Memory, we train our two-stream network with totally seven different feature maps instead of eight, by taking zooming out 3 times on the stream with branching stride=2, and 2 times on the stream with branching stride=3.

5.2. Inference

During the inference stage, we process a image in one shot and extract windows at different scaled feature maps similar to the training stage. To boost up the performance, we separate the confidence decoding and mask decoding from head and apply them differently. Concretely speaking, we first use the confidence decoder to predict the confidence score of each window, and then only select the top k confident windows to decode their object segmentation mask.

As we learned our residual neck to scale down feature maps smoothly and keep feature well-calibrated, we could add or reduce the number of neck component during inference. As mentioned before, we train our network with seven scales of feature maps and test on a full two-stream model, with eight scales of feature maps, and still obtain excellent performance. This enable us to make trade-off between the effectiveness and efficiency, via adjusting input image size and the number of differently scaled feature.

To further clarify our implementation details, we made our code public available on: <https://github.com/voidrank/FastMask>.

6. Experiments

We analyze and evaluate our network on MS COCO benchmark, which contains 80k training images and a total

Method	Body Net	Box Proposals			Segmentation Proposals		
		AR@10	AR@100	AR@1k	AR@10	AR@100	AR@1k
MCG	-	10.1	24.6	39.8	7.7	18.6	29.9
DeepMask [17]	VGG	15.3	31.3	44.6	12.6	24.5	33.1
DeepMaskZoom [17]	VGG	15.0	32.6	48.2	12.7	26.1	36.6
DeepMask* [18]	Res39	18.0	34.8	47.0	14.1	25.8	33.6
SharpMask [18]	Res39	19.2	36.2	48.3	15.4	27.8	36.0
SharpMaskZoom [18]	Res39	19.2	39.0	53.2	15.6	30.4	40.1
InstanceFCN [5]	VGG	-	-	-	16.6	31.7	39.2
FastMask+two streams	Res39	22.6	43.1	57.4	16.9	31.3	40.6
FastMask+two streams	Pva	24.1	43.6	56.2	17.5	30.7	39.0

Table 3. Object segment proposals result on COCO validation set for box and segmentation proposals. Note that we also report the body network for each corresponding method.

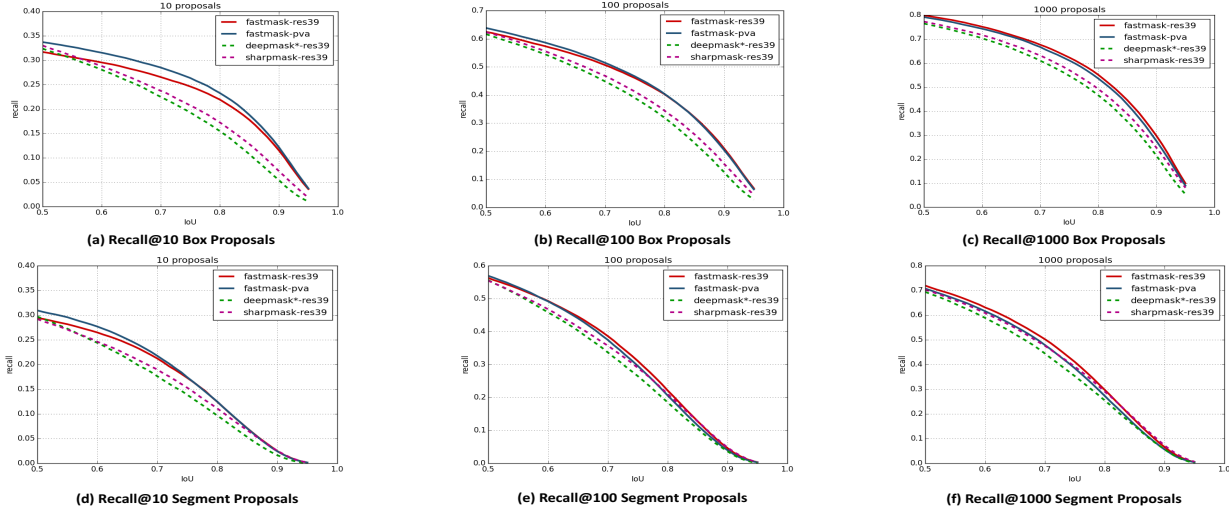


Figure 6. Proposal recall curve. (a-c) show detailed box proposal recall while (d-e) show detailed segment proposal recall.

of nearly 500k instance annotations. Following the experiment setting of [17, 18, 5], we report our result on the first 5k COCO validation images. We use another non-overlapped 5k images for validation.

Metrics. We measure the mask accuracy by Intersection over Union (IoU) between predicted mask and ground truth annotation. As average recall correlates well with object detector performance [13], we summarize the object proposals accuracy using Average Recall (AR) between IoU 0.5 and 0.95 for a fixed number N of proposals, denoted as "AR@ N ". Our results are measured with N equals to 10, 100 and 1000.

Scales. As COCO dataset contains objects in a wide range of scales, there is a more fine-grained evaluation that measures metrics with regards to object scales. In practice, the object are divided into three groups according to their pixel area a : small($a \leq 32^2$), medium($32^2 < a \leq 96^2$), large($a > 96^2$). In our experiment, we denote the metrics for different scales by adding a superscript S , M , L respectfully.

Methods. By default, we compare our method with recent state-of-the-art methods for segment proposal, includ-

ing *DeepMask* [17], *SharpMask* [18] and *InstanceFCN* [5]. Note that we also provide results of a revised DeepMask architecture from [18], denoted as *DeepMask**. Different from original DeepMask, it is implemented based on 39-layer residual net with a revised head component. These methods not only achieves good AR in segment proposal but also provide good efficiency during inference.

Our network is general and could be plug-in to different body networks. In our experiments, we adopt 39-layer Residual Net [10] for best accuracy as well as fair comparison and PvaNet [12] for best efficiency.

6.1. Comparison with State-of-the-art Methods

Table 3 compares the performance of our FastMask to other state-of-the-art methods. We report results on both box and segment proposals (by extracting a tight bounding boxes from mask proposals). Here we do not include the SharpMaskZoom² result because they take images with extra scales ($2^{1/2}$ larger) as input to obtain superior performance.

We compare our two-stream FastMask with all those im-

Method	Scales	AR@10	AR@100	Speed
DeepMask* [18]	8	14.3	27.3	0.45s
DeepMask* [18]	4	11.3	22.2	0.24s
FastMask	8	16.9	31.3	0.25s
FastMask	4	13.3	26.6	0.14s

Table 4. Trade-off between scale density and performance.

age pyramid method as our single stream network do not have a dense coverage of feature scales comparing to those image pyramid methods. To address the influence of feature scale density to performance as well as speed, we conduct a separate experiments in section 6.2.

Quantitative evaluation. According to Table 3, we outperforms all state-of-the-art method in bounding-box proposal by a large margin and obtain very competitive results with segmentation proposals(outperform all methods on AR@10 and AR@1k, and show competitive performance on AR@100). It is worth noting that our two stream network significantly improve the box proposal quality comparing to all other methods. Our two stream FastMask model with 39-layers Resnet achieve approximately 18%, 11%, 8% relative improvement on AR@10, AR@100, AR@1k metrics respectively, over previous best SharpMask model. In order to give a better picture of our proposal quality, we plot the recall versus IoU threshold for different of segmentation proposals in COCO dataset as Figure 6. There is a clear gap in the plot, which indicates that FastMask produce better mask quality overall.

While obtaining superior performance, our method also yields better efficiency than all those image pyramid based approaches. We did some control experiments and report the speed/performance in section 6.2.

Qualitative visualization. We visualize some results in Figure 7 showing exemplars on which our method improves over baseline predictions (More visualization results in supplementary material). Generally, we could observe that our method is more invariant to noisy background. Note that FastMask does not perform any refinement on mask, it is possible that adding mask refinement could further boost up mask quality.

6.2. Efficiency Study

In this section, we evaluate two threads to support our argument that FastMask is better than image pyramid methods in terms of inference efficiency and performance. On the first thread, we provide control experiment results on DeepMask and SharpMask, with restriction on the scale density of their image pyramids. We construct a fair environment that both these methods and our method take equivalently many scales and evaluate both inference speed and performance. On the other thread, we provide the performance and speed of state-of-the-art methods and compare our best model as well as fastest model.

Trade-off scale density with speed. We conduct a fair

Method	Body Net	AR@10	AR@100	AR@1k	Speed
DeepMask [17]	VGG	12.6	24.5	33.1	1.60s
DeepMask* [18]	Res39	14.1	25.8	33.6	0.46s
SharpMask [18]	Res39	15.4	27.8	36.0	0.76s
SharpMaskZoom [18]	Res39	15.6	30.4	40.1	~1.5s
InstanceFCN [5]	VGG	16.6	31.7	39.2	1.50s
FastMask-acc	Res39	16.9	31.3	40.6	0.26s
FastMask-fast	Pva	17.2	29.4	36.4	0.07s

Table 5. Speed Study with state-of-the-art methods.

study to analyze the trade-off by degrading scale density. In the DeepMaskZoom* and SharpMaskZoom, they inference on images scaled from $2^{\wedge}[-2.5, -2.0, -1.5, -1.0, -0.5, 0, 0.5, 1]$ to obtain superior performance on a diverse range of object segments. This is similar to the setting of two-stream network, where we take a image up-sampled by two. To improve the inference efficiency, we trade-off scale density by simply making our network one-stream with out re-training, which is identical to reduce the scale density of DeepMaskZoom* and SharpMaskZoom to $2^{\wedge}[-2.5, -1.5, -0.5, 0.5]$.

Figure 4 illustrates the performance degrade and efficiency increase with scale density trade-off. We measure only AR@10 and AR@100 as scale density degradation reduces total proposal number. This control experiments are tested using NVIDIA Titan X GPU. We do multiple run and average the run time to obtain final performance. Our methods achieve to preserve the best performance while increase the inference speed by almost $2\times$. Note that we could directly train a network with this reduced scale density to boost up performance.

Speed evaluation. We evaluate the inference speed of all state-of-the-art methods. We report efficiency results on two variant of our models – our best performance model (FastMask-acc) and fastest model (FastMask-fast). Our best performance model take a two-stream structure with 39-layer ResNet. Our fastest model take a one-stream structure with PvaNet [12], which is light-weight and fast.

Figure 5 compare our best and fastest model with other networks. Our best mode produces superior result while preserving good efficiency. Our fastest model trade-off slightly in performance and obtain almost real-time performance (~ 13 FPS by NVIDIA Titan X GPU).

7. Conclusion

In this paper we present an innovative framework, *i.e.* FastMask, for efficient segment-based object proposal. Instead of building pyramid on input image, FastMask learns to encode feature pyramid by a neck module, which enables multi-scale training and efficient one-shot inference. Furthermore, we augment FastMask with a scale-tolerance head to reduce the impact of background noises, resulting in a significant improvement on segmentation accuracy. On



Figure 7. Visualization of the object candidate segmentation results on sample MS COCO images. We compare our FastMask with DeepMask* [18] and SharpMask [18]. We also show the origin images and the ground-truth annotations for reference.

MS COCO benchmark, FastMask outperforms all state-of-the-art segment proposal methods in average recall while keeping two times faster at least. More impressively, with a slight tradeoff in accuracy, FastMask can segment objects in near real time (~ 13 fps) at 800×600 resolution images. As an effective and efficient segment proposal method, FastMask is believed to have great potentials in other vision-related tasks.

References

- [1] P. Arbeláez, J. Pont-Tuset, J. T. Barron, F. Marques, and J. Malik. Multiscale combinatorial grouping. In *CVPR*, 2014. 1, 2
- [2] S. Bell, C. L. Zitnick, K. Bala, and R. Girshick. Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks. In *CVPR*, 2016. 1, 2
- [3] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. In *ICLR*, 2015. 2
- [4] M.-M. Cheng, Z. Zhang, W.-Y. Lin, and P. Torr. Bing: Binarized normed gradients for objectness estimation at 300fps. In *CVPR*, 2014. 2
- [5] J. Dai, K. He, Y. Li, S. Ren, and J. Sun. Instance-sensitive fully convolutional networks. In *ECCV*, 2016. 1, 2, 7, 8
- [6] J. Dai, K. He, and J. Sun. Instance-aware semantic segmentation via multi-task network cascades. In *CVPR*, 2016. 1, 2
- [7] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov. Scalable object detection using deep neural networks. In *CVPR*, 2014. 2
- [8] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 1, 6

- [9] R. Gokberk Cinbis, J. Verbeek, and C. Schmid. Segmentation driven object detection with fisher vectors. In *CVPR*, 2013. 1
- [10] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 3, 4, 7
- [11] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *ECCV*, 2016. 4
- [12] S. Hong, B. Roh, K.-H. Kim, Y. Cheon, and M. Park. Pvanet: Lightweight deep neural networks for real-time object detection. *arXiv preprint arXiv:1611.08588*, 2016. 7, 8
- [13] J. Hosang, R. Benenson, P. Dollár, and B. Schiele. What makes for effective detection proposals? *IEEE T-PAMI*, 2016. 7
- [14] P. Krähenbühl and V. Koltun. Geodesic object proposals. In *ECCV*, 2014. 1, 2
- [15] W. Kuo, B. Hariharan, and J. Malik. Deepbox: Learning objectness with convolutional networks. In *ICCV*, 2015. 2
- [16] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014. 2
- [17] P. O. Pinheiro, R. Collobert, and P. Dollar. Learning to segment object candidates. In *NIPS*, 2015. 1, 2, 3, 4, 5, 7, 8
- [18] P. O. Pinheiro, T.-Y. Lin, R. Collobert, and P. Dollár. Learning to refine object segments. In *ECCV*, 2016. 1, 2, 4, 7, 8, 9
- [19] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*, 2015. 2, 6
- [20] E. Shelhamer, J. Long, and T. Darrell. Fully convolutional networks for semantic segmentation. *IEEE T-PAMI*, 2016. 2
- [21] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 3, 4, 5
- [22] K. E. Van de Sande, J. R. Uijlings, T. Gevers, and A. W. Smeulders. Segmentation as selective search for object recognition. In *ICCV*. IEEE, 2011. 1, 2
- [23] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. In *ICLR*, 2016. 2
- [24] Z. Zhang, J. Warrell, and P. H. Torr. Proposal generation for object detection using cascaded ranking svms. In *CVPR*, 2011. 1, 2
- [25] C. L. Zitnick and P. Dollár. Edge boxes: Locating object proposals from edges. In *ECCV*, 2014. 2